〈経営研究　第11巻　第1号　平成9年7月〉

パソコンによる2連字についてのエントロピー計算実験

横井右門

〈キー・ワード〉
- 2次元と1次元（2nd dimension and 1st dimension）
- エントロピーの定義（definition of entropy）
- メモリー・サイズ（storage size）
- 大量データ（large-file text data）
- 32ビット・マシン（32-bit cpu）
- 改行とスペース（returns and spaces）

1文字についてのエントロピーよりも2連字についてのエントロピーは，小さい。3連字についてのエントロピーは，さらに小さい。8連字についてのエントロピー2.35が計算された最高記録であると言われている。また，15連字のあたりで1.3に収束するとも言われている。しかし，2.35というエントロピーが求められたのは1955年以前である[1]。どの程度の計算手段を使ったのか，データ量はどれほどだったのか。大いに疑問がある。データの出所，アルゴリズムをはっきりさせ，4.6メガ・バイトのデータによって追試験を行い，より桁数の多いエントロピーを求めようというのが，本稿を含めた一連の実験の目的である。

1．エントロピー

エントロピーという言葉は，熱力学でも，統計力学でも使われている。また，情報理論でもエントロピーが使われている。最近のエントロピーという言葉が使われた例を紹介する。

早坂　暁

「そして，同一均質社会ほど，嫉妬エネルギーが増大するのだ。いわばエントロピー増大だ。」(2)

石原慎太郎

「エントロピーは人間たちにこの世の有限について教えたといわれるが，ほとんどの人間にはその実感も乏しく人々は未だに地球や宇宙の広大さの迷妄から覚めることはない。」(3)

西部　邁

「エントロピーの増大こそがエントロピーの減少なのだといわれても私にはまったくの珍糞漢である。」(4)

これらの「エントロピー」が，熱力学・統計力学のそれなのか，情報理論のそれなのか，判断するのは難しい。それに比べ，「複雑系」でのエントロピーの使い方は明確である(5)。

また，エントロピーが概念であって，計算できるものではないという認識も，散見される。本実験の発端はそういう誤解を具体的な例証によって排除したいという点にもあった。

しかし，この2種類のエントロピーについては，三十年以上も前に明確な見解が確立している。J．R．ピアースが次のように言っている。

この安直だが誤りを招きやすい観念は専門家の間にさえたくさんの混乱をひき起こした。本当はコミュニケーション理論は電気通信の分野のある種の

問題を解くために生まれたのである。そのエントロピーは，統計力学のエントロピーとの数学的類似のためにエントロピーと名づけられたのである。このエントロピーが主に関係する問題は，統計力学が取り組む問題とはまったく異なるものである[6]。

2．エントロピーの定義

1文字についてのエントロピーの定義はつぎのとおりである[7][10]。

言語がn個の記号（番号1からnで区別する）で表現され，それぞれの出現確率がP(n)であるとき，次の式で定義される。

$$\text{エントロピー} = \sum_{i=1}^{n} P(i)\log_2 P(i)$$

ただし，この値は－になるので，符号を逆転させる。このエントロピーが重要なのは，この値は実は，分析の対象である言語において，1つの記号を送るのに必要な平均ビット数を表しているからである。

本稿は2文字が続く2連字についてのエントロピーを扱うので，式は二次元を扱うΣが二重になる少しややこしい式になる。ASCII96字について1次のエントロピーは，上の式で計算される。上の定義のnを96とすればよい。ここでASCII 2文字で1つの記号と考えれば，2連字のときは，9216個の記号で表現されることになる。9216は96の自乗である。JIS第2水準まで考えた日本語よりも少し多いだけの記号数にすぎない。定義のnを9216とするだけである。こうすれば，2連字だからといっても，上の1文字についてのエントロピーの定義でまにあってしまう。3連字のときでも，96の3乗は884,736だからnを884,736にするだけである。Microsoft Cは16ビット・コンパイラーであり，64K以内の配列しか扱えなかったが，Visual C++では，はるかに大きい配列を扱うことができるので問題はない。

3．行末の改行キーの扱い

シャノンは，英字26字とスペースの27文字についてエントロピーを計算したが[1][6][7][10]電報の場合はこれでよいが，小説等の書籍については問題がある。水平右方向に無限の長さがある紙のページがあればよいが，そんなわけには行かない。ペーパーバックの場合は80字より小さい限界がある。ハードカバーでも100字未満である。改行キーはASCII96文字の範囲外である。行末の次の行の最左端から次の語の最初の文字が始まる。したがって行末の単語と次の行の先頭の単語の間にはスペースがないことになる。かりに1行60字だとすると，60分の1の誤差が生じることになる。有効数字の大きいエントロピーを計算するのが実験の目的であるから，かなりの問題である。そこで，データのテキスト・ファイルを簡単なプログラムを通して，各行末にスペースを1個追加した。

4．採用したデータ

データとして次の8冊のペーパーバックを使った。

M1：Anne McCaffrey: The Crystal Singer
M2：Anne McCaffrey: Crystal Line
P1：Robert B. Parker: Pastime
P2：Robert B. Parker: The Catskill Eagle
C1：Tom Clancy: The Hunt for Red October
C2：Tom Clancy: Op-Center
Q1：A. J. Quinnell: In the Name of the Father
Q2：A. J. Quinnell: The Blue Ring

磁気媒体への変換は，全て手作業入力である。スキャナー・システムを採

用せずに，あえて手作業で行ったのは，手元に性能の良いスキャナー・システムがなかったせいもあるが，次の理由による。

かりに99.9パーセントの正読率のスキャナー・システムが存在したとしてもM1のThe Hunt for Red Octoberは，約97万バイトあり，479ページであるから1ページに約2文字の間違いがあることになる[8]。長年校正作業に携わった経験から，1ページに2個間違いがあった場合も，1ページに20個間違いがあった場合でも手間はほとんど変わらない。

それにブラインド・タッチで入力すると，小説を読む楽しみが付随してくる。SF，ハードボイルドおよび冒険小説を選んだのもそのためである。大量のキーボード入力の苦痛が軽減される効果がある。

M1の原書には少なくとも20箇所の間違いがあるが，入力間違いはそれよりも少ないはずである。

5．頻度を数えるプログラムと解説

まず，79行という短いプログラムなので，そのまま載せる。

```
0001: /* テキスト･ファイルの2連字別頻度 KILLA110.C */
0002: #include <stdio.h>
0003: #include <time.h>
0004: #include <math.h>
0005:
0006: void main(void)
0007: {
0008:     long nowtime, start, finish;
0009:     int  ct, c[96][96], i, j, column_cut, valid_cnt;
0010:     char  infile_name[50], left_chara, right_chara, trace,
0011:         outfile_name[50];
0012:
0013:     FILE*fp, fs;
0014:     for(i=0; i<96; i++)
0015: {
0016:       for(j=0; j<96; j++)
0017:       {
```

```
0018:         c[i][j]=0;
0019:       }
0020: }
0021:
0022:     trace = ' ';
0023:
0024:     printf("(KILLA110.C)  ");
0025:     time(&nowtime);
0026:     printf("%s ¥n", ctime(&nowtime));
0027:
0028:     printf("machine: ");
0029:     gets(infile_name);
0030:     printf("%s¥n", infile_name);
0031:     printf("Title: ");
0032:     gets(infile_name);
0033:     printf("%s ¥n", infile_name);
0034:     printf("file name: ");
0035:     scanf("%s", infile_name);
0036:     printf("%s¥n¥n", infile_name);
0037:     column_cnt=valid_cnt=0;
0038:     fp = fopen(infile_name, "r");
0039:
0040:     time(&start);
0041:
0042:     left_chara=getc(fp);
0043:     i=left_chara-32;
0044:
0045:     if(fp != NULL)
0046:     {
0047:        while((right_chara = getc(fp)) != EOF)
0048:        {
0049:           j = right_chara-32;
0050:
0051:           if(32<=right_chara && right_chara<=127)
0052:           {
0053:              ct=ct+1;
0054:              c[i][j]=c[i][j]+1;
0055:              left_chara=right_chara;
0056:              i=left_chara-32;
0057:           }
0058:        }
0059:     }
0060:     time(&finish);
0061:     printf("time = %7.0f ¥n", difftime(finish, start));
```

```
0062:     printf("Enter outfile name | n");
0063:     scanf("%s", outfile_name);
0064:     fs=fopen(outfile_name, "w");
0065:     fprintf(fs, "%c%c%7d¥n", 0xff, oxff, ct);
0066:
0067:     for(i=0; i<96; i++)
0068:     {
0069:       for(j=0; j<96; j++)
0070:       {
0071:         fprintf(fs, "%c%c%7d¥n", i+32, j+32, c[i][j]);
0072:         column_cnt ++;
0073:         valid_cnt ++;
0074:       }
0075:     }
0076:
0077:     printf("valid couples=%d¥n", valid_cnt);
0078:
0079: }
```

解説

IBMにHIPO(Hierarchy plus Input Process Output)という文書化の技法があるが，そのprocessの部分だと了解していただきたい。括弧内は行番号である。

① 二次元配列をゼロクリアする。(14～20)

② プログラム名と実行開始日時を表示する。(24～26)

③ 実行コンピュータ名を表示する。(28～30)

④ テキスト・データの書名を表示する。(31～33)

⑤ テキスト・データのドライブとファイル名を入力・表示する。(34～36)

⑥ ファイルをオープンする。(38)

⑦ 主要反復手続の開始時刻の記憶(40)

⑧ 第一文字を読む。32を引いて配列の行番号とする。(42～43)

⑨ 次の字を読みEOFでなければ以下58行までを実行する。(47)

⑩ 文字から32を引いて配列の列番号とする。(49)

⑪ 文字カウンターと該当する配列要素に1を加える。

⑫ 先行する文字に転記し，列番号を計算する。(55～56)

⑬ 経過時間の表示。（60～61）

⑭ 出力ファイルをオープンする。（62～64）

⑮ 2バイトを16進FFFFとし，入力ファイルの文字数を第2項目とし，1レコード出力する。（65）

⑯ 行列番号からそれぞれ32を引き，先頭2バイトに転記し，配列の値を第2項目として順次レコードを出力する。（67～75）

6．2連字エントロピーの計算

前項の手続で出力されたファイルを，MS-DOSのSORTコマンドで，頻度について逆順で整列させる。

先頭のレコードの第2項目は，総文字数になるから，それを分母にし，つぎのレコードの第2項目すなわち頻度を除すれば，出現確率が計算できる。定義に従い底2の出現確率の対数との積を累計すれば，エントロピーが求められる。

その結果を以下に示す。

書籍コード	1次エントロピー	二次エントロピー
M1	4.439092	3.938997
M2	4.426147	3.968438
P1	4.445120	3.975126
P2	4.466666	3.995353
C1	4.457179	4.029204
C2	4.465101	4.005841
Q1	4.417990	3.968828
Q2	4.387838	3.951569
全8冊	4.457096	4.013684

７．頻度順データの特徴

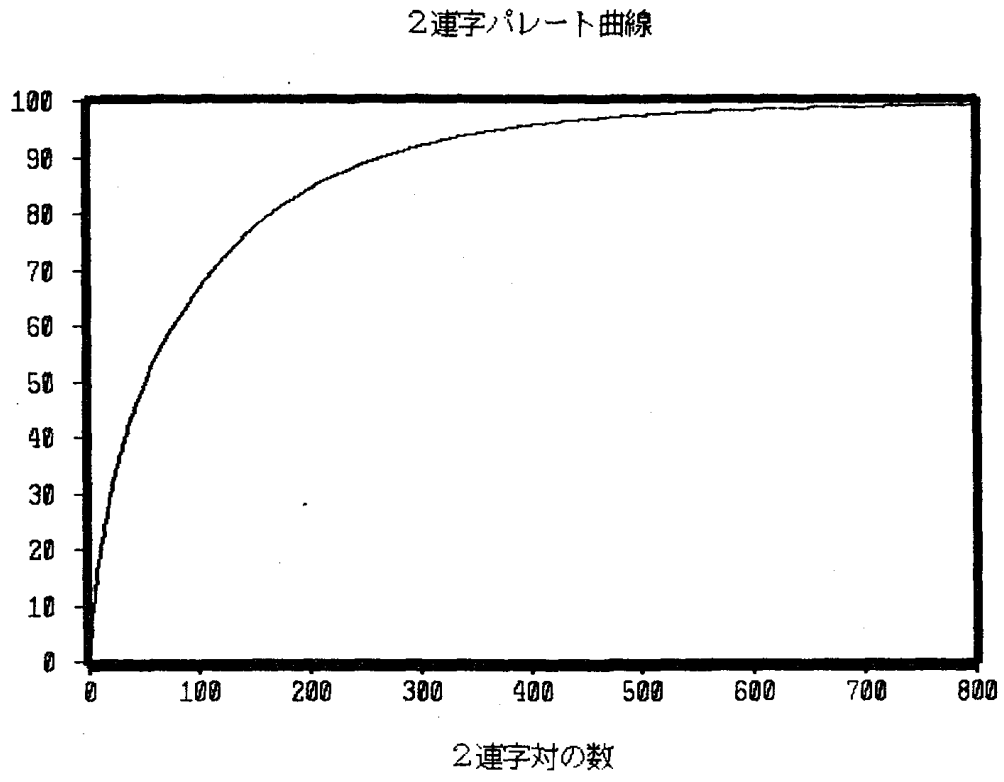
まず先頭の99対の２連字を示す。

左から番号，２連字，その対の頻度，相対頻度の順にしめしてある。

さらに続けてそのパレート曲線のグラフも示すが，実際に発生した2,761対のうち，第166対で80パーセントに達し，第266対で90パーセント，第783対で99パーセントに達する。一見しただけで，きれいなパレート曲線を描いていることが判断できる。

4518749

1	e	137561	3.0442	34	ng	30148	0.6672	67	de	16196	0.3584
2	t	107948	2.3889	35	c	29391	0.6504	68	r	15470	0.3424
3	he	101096	2.2373	36	as	28603	0.6330	69	ad	15307	0.3387
4	d	87012	1.9256	37	or	26400	0.5842	70	ai	15129	0.3348
5	th	84795	1.8765	38	hi	26188	0.5795	71	ut	14964	0.3312
6	a	75738	1.6761	39	ar	26119	0.5780	72	co	14954	0.3309
7		74442	1.6474	40	f	25751	0.5699	73	be	14821	0.3280
8	t	69726	1.5430	41	te	25317	0.5603	74	ra	14622	0.3236
9	s	65088	1.4404	42	st	24656	0.5456	75	ri	14564	0.3223
10	in	60265	1.3337	43	is	24293	0.5376	76	id	14463	0.3201
11	s	57634	1.2754	44	it	23780	0.5263	77	sh	14384	0.3183
12	er	56013	1.2396	45	m	23713	0.5248	78	h	14347	0.3175
13	.	53621	1.1866	46	es	22229	0.4919	79	el	14295	0.3163
14	an	53605	1.1863	47	le	22004	0.4869	80	li	14168	0.3135
15	h	53286	1.1792	48	g	21681	0.4798	81	om	14122	0.3125
16	n	49107	1.0867	49	a	21512	0.4761	82	″	13948	0.3087
17	w	47664	1.0548	50	ve	21340	0.4723	83	″	13797	0.3053
18	ed	45253	1.0014	51	ll	21268	0.4707	84	n	13750	0.3043
19	re	44119	0.9764	52	d	20869	0.4618	85	il	13749	0.3043
20	o	40305	0.8920	53	ne	20840	0.4612	86	ur	13456	0.2978
21	r	38474	0.8514	54	al	20681	0.4577	87	no	13329	0.2950
22	nd	37857	0.8378	55	wa	20271	0.4486	88	g	13146	0.2909
23	,	37598	0.8320	56	nt	20190	0.4468	89	la	13066	0.2892
24	ou	36874	0.8160	57	ea	19932	0.4411	90	e	12920	0.2859
25	on	35790	0.7920	58	p	19660	0.4351	91	ta	12847	0.2843
26	ha	34630	0.7664	59	f	19527	0.4321	92	ce	12660	0.2802
27	at	33914	0.7505	60	se	19448	0.4304	93	ee	12536	0.2774
28	o	32479	0.7188	61	ti	18788	0.4158	94	I	12491	0.2764
29	en	31753	0.7027	62	me	18656	0.4129	95	ac	12423	0.2749
30	to	31664	0.7007	63	l	18451	0.4083	96	oo	12343	0.2732
31	i	31106	0.6884	64	of	18022	0.3988	97	ot	12313	0.2725
32	y	31068	0.6875	65	l	17674	0.3911	98	ow	12271	0.2716
33	b	30340	0.6714	66	ro	16634	0.3681	99	si	12262	0.2714



8．結論

1次エントロピーよりも2次エントロピーは，どの小説についても，小さくなっている。全8冊の合計の場合でも同じことがいえる。

96の自乗である9216が対の論理的可能性であるが，実際にはその29パーセントにあたる2,761対しか発生していない。

9．今後の計画

さらに3次エントロピー，4次エントロピーの計算にとりかかる。今回の実験結果からみて，3次エントロピーの計算は，時間がかかるだけで，それほど困難ではないことが予想できる。4次エントロピーは，ほんの少し工夫が必要であるが，時間がさらにかかるだけで，かつて記憶容量とソフトのバグのせいで，一ヶ月かかっても，結局SORTが終了しなかった失敗に比べれば，楽なものである。

6次エントロピーの計算が終了した段階で，語についてのエントロピーに取り組みたい。英語の単語長は，タイピングでは，5ということになっているからである。

その次には，語による相関分析を行い，作者の特定が可能か否かの計算にとりくむ。

参考文献

(1) 南　敏：情報理論，産業図書（1988）
(2) 早坂　暁：花へんろ通信，週刊新潮97.1.2，新潮社（1997）
(3) 石原慎太郎：国家なる幻影，文芸春秋97年2月号，文芸春秋社（1997）
(4) 西部　邁：改革のエントロピー，正論97年2月号，産経新聞社（1997）
(5) M. ミッチェル. ワールドロップ：複雑系，新潮社（1996）
(6) J.R. Pierce: Symbols, Signals and Noise, Harper & Row (1961)
(7) 前川　守：文章を科学する，岩波書店（1996）
(8) 岡田　毅：実践コンピュータ英語学，鶴見書店（1995）
(9) Kernighan, B.W. and P.J. Plauger: The Elements of Programming Style, McGraw-Hill (1974)
(10) C.E. Shannon: The Mathematical Theory of Communication, Bell System Technical Journal, July and October (1948)